Gumpy: A Python Toolbox Suitable for Hybrid Brain-Computer Interfaces

Zied Tayeb [1,2], Nicolai Waniek [1], Juri Fedjaev [1], Nejla Ghaboosi [3], Leonard Rychly [1], Christian Widderich [1], Christoph Richter [1], Jonas Braun [1], Matteo Saveriano [4], Gordon Cheng [2] and Jörg Conradt [1]

[1] Neuroscientific System Theory, Department of Electrical and Computer Engineering, Technical University of Munich, Germany

[2] Institute for Cognitive Systems, Technical University of Munich, Germany

[3] Integrated Research, Sydney, Australia

[4] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany

E-mail: `zied.tayeb@tum.de`

Abstract. Objective. The objective of this work is to present *gumpy*, a new free and open source Python toolbox designed for hybrid brain-computer interface (BCI). **Approach**. Gumpy provides state-of-the-art algorithms and includes a rich selection of signal processing methods that have been employed by the BCI community over the last 20 years. In addition, a wide range of classification methods that span from classical machine learning algorithms to deep neural network models are provided. Gumpy can be used for both EEG and EMG biosignal analysis, visualization, real-time streaming and decoding. **Results**. The usage of the toolbox was demonstrated through two different offline example studies, namely movement prediction from EEG motor imagery, and the decoding of natural grasp movements with the applied finger forces from surface EMG (sEMG) signals. Additionally, *gumpy* was used for real-time control of a robot arm using steady-state visually evoked potentials (SSVEP) as well as for real-time prosthetic hand control using sEMG. Overall, obtained results with the *gumpy* toolbox are comparable or better than previously reported results on the same datasets. **Significance**. Gumpy is a free and open source software, which allows end-users to perform online hybrid BCIs and provides different techniques for processing and decoding of EEG and EMG signals. More importantly, the achieved results reveal that *gumpy*'s deep learning toolbox can match or outperform the state-of-the-art in terms of accuracy. This can therefore enable BCI researchers to develop more robust decoding algorithms using novel techniques and hence chart a route ahead for new BCI improvements.

Keywords: Hybrid Brain-Computer Interfaces, Python toolbox, Deep Learning, EEG, EMG.

Submitted to: *J. Neural Eng.*

1. Introduction

Paralyzed people wish to control assistive devices such as wheelchairs, spellers, prosthetics, or exoskeletons in order to improve their quality of life and ensure their independence [1]. One way to infer their desired actions is to measure their cortical activity, for instance by functional magnetic resonance imaging (fMRI), magnetoencephalography (MEG), electrocorticography (ECoG), or electroencephalography (EEG) and subsequently decode the intended movement from the measurements. Of these methods, EEG has become the most frequently used technique for BCIs, because it is non-invasive and comparably inexpensive. Although BCI technology has seen significant improvements over the last few years [2, 3], it still lacks reliability and accuracy. Hybrid BCIs in general [4], and particularly those which combine EEG and EMG signals are promising significant improvements [5]. Despite the successful multidimensional EEG-based BCI control achieved using simple classifiers [6, 3], reliable decoding of complex movements from brain signals is still challenging and requires advanced algorithms [7]. Recent developed techniques such as deep neural networks [8] could represent a promising solution to develop more robust decoding algorithms [7]. In order to make such algorithms readily available to a wide BCI community we developed *gumpy*, a Python library along with well documented application examples that we introduce in this paper. Gumpy is an easy-to-use, robust, and powerful software package for EEG and EMG signal analysis and decoding that tightly incorporates different recording paradigms, essential signal processing techniques, and state-of-the-art machine learning algorithms. Gumpy can be used for offline as well as for online processing of electrophysiological signals. Several similar BCI software packages exist and are widely used by the community [9]. Gumpy is free of charge, permissively licensed and written in Python, an open source programming language that is not only backed by an extensive standard library, but also by vast scientific computing libraries. Moreover, it is widely used by many machine learning experts, engineers and neuroscientists. Gumpy offers users the opportunity to reproduce results previously achieved by other BCI researchers through implementing a wide range of signal processing and classification methods for time series signal analysis. Furthermore, the toolbox features several deep learning models such as deep convolutional neural networks (CNN) [10], recurrent convolutional neural networks (RCNN) and Long Short-Term Memory (LSTM) [11]. Those approaches have hitherto been rarely investigated by BCI researchers [12] and to the best of our knowledge no existing BCI software integrates similar techniques. This paper introduces the basic concept of *gumpy*, its main features and three successful BCI applications. The remainder of this paper is structured as follows: Section 2 provides an overview of related work and reviews existing BCI toolboxes. Section 3 describes *gumpy*'s design and its main features and functions. Section 4 and 5 demonstrate, respectively, the basic offline and online usage of *gumpy* on different tasks, such as motor imagery (MI) movements decoding from EEG, and the prediction of hand gestures from sEMG. Finally, Section 6 enumerates

gumpy's strengths and weaknesses and proposes possible future developments.

2. Related work

This section provides an overview of the most widely-used open source BCI platforms for research and highlights the distinctive features of *gumpy* with respect to them. Table 1 summarizes their main functions and limitations. References [9, 13] provide a more comprehensive survey. The discussion focuses on a particular feature set that we deem essential for the successful development of future hybrid BCI systems.

2.1. BCILAB

BCILAB [14] is among the earliest publicly available BCI software packages for research purposes. It is a free, open source toolbox developed in Matlab. BCILAB is built to emulate the plugin concept where various components can be added "on the fly" during the runtime. BCILAB was designed as an extension of EEGLAB [15] to support both offline and online analysis of electrophysiological signals. Besides various feature extraction methods and experimental paradigms supported by the toolbox, an end-user can choose between three different classifiers (Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and Support Vector Machine (SVM)). In addition, BCILAB obliges users to design their scripts in Matlab [14].

2.2. BCI2000

BCI2000 [16] is an open source and free of charge BCI software package developed in 2000 to advance real-time BCI systems. It includes different modules such as data acquisition, signal processing and stimulus presentation. The toolbox is written in C++ and does not directly support other programming languages such as Matlab or Python, so in this regard it is difficult to extend and integrate with other toolboxes. Furthermore, some important processing methods such as discrete wavelet transform and some classification techniques such as deep learning are not included [16].

2.3. MNE

MNE is an open source Python package for MEG/EEG data analysis. MNE implements a wide range of functions for time-frequency analysis and connectivity estimation as well as simple decoding algorithms [17]. Similar to *gumpy*, it is built on top of widely used scientific computing libraries such as NumPy [18], SciPy [19], pandas and scikit-learn [20]. Moreover, MNE offers functions for neuroimaging data interpretation such as fMRI analysis. Despite recent developments, the toolbox still lacks some important functions and methods, such as common spatial pattern algorithm (CSP) [21] and various popular machine learning classifiers.

2.4. Wyrm

Wyrm [22] is an open source BCI package written in Python. The toolbox implements several functions for processing and visualization of electrophysiological data such as EEG and ECoG signals. Moreover, Wyrm is suitable for both offline processing and real-time applications. Furthermore, the toolbox integrates Mushu [23] a free software for signals acquisition, and Pyff [24], which is a framework for BCI feedback applications.

2.5. OpenViBE

OpenViBE [25], another open source BCI platform, is designed in a modular fashion and incorporates an elegant graphical user interface for novice users. Moreover, it provides a wide range of signal processing techniques and supports many acquisition and BCI paradigms such as P300 [26, 27] and SSVEP [28]. One of OpenViBE's advantages with respect to the previously mentioned toolboxes is that it can be scripted using both LUA and Python. In addition, it offers a direct interface to Matlab. OpenViBE currently provides three classifiers: LDA, SVM as well as a combined classifier for a multi-class problem classification.

Table 1: General Overview of Existing BCI toolboxes.

Software platform	**Programming language**	**Features**
BCILAB	Matlab	Wide range of algorithms Well-designed GUI
BCI2000	C++	Simple and Robust Wide usage by BCI community Modular programming
MNE	Python	EEG, MEG and fMRI data analysis Good documentation
Wyrm	Python	EEG and ECoG signals Real-time capabilities Integration with other platforms
OpenViBE	LUA, Python	Modular API Supports many acquisition devices
Gumpy	Python	Hybrid BCI Real-time capabilities Offline and online analyses Deep learning toolbox

2.6. Distinctive features of gumpy

Despite the tremendous number of features that current BCI toolboxes offer, they still exhibit some limitations [13] such as a lack of important processing and classification methods, limited real-time performance, or lack of experimental paradigms to conduct online BCI experiments. More importantly, none of the existing packages combine classic machine learning algorithms and deep learning techniques for signals decoding. However,

gumpy covers a wide range of classification methods including several machine learning classifiers, voting classifiers, feature selection algorithms and different deep learning architectures such as LSTM, CNN and RCNN. Additionally, we provide many showcase examples of the usage of *gumpy* for EEG and EMG signals decoding, thereby facilitating the design of hybrid BCI systems. Furthermore, *gumpy* integrates different experimental paradigms for motor imagery, EMG, SSVEP, EEG reach-to-grasp movements recording and hybrid BCI, which can be easily used and extended by end-users. Importantly, *gumpy* supports offline analysis as well as online BCI applications. With the lab streaming layer (LSL) [29] for data acquisition, the provided experimental paradigms for biosignals recording and *gumpy* package for EEG and EMG data processing and decoding, we envision *gumpy* to be a suitable toolkit for conducting online hybrid BCI experiments.

3. Gumpy toolbox: design, main functions and features

3.1. General overview of gumpy's modules

Gumpy comprises six modules for plotting, processing, and classification of EEG and EMG signals. Moreover, *gumpy* incorporates different deep learning models and experimental recording paradigms. This section provides a condensed overview of *gumpy*'s modules and its main functionality, which are summarized in Figure 1. Some of the modules are described in more detail in the following section on exemplary use-cases. Particularly, subsection 3.3 covers the available deep learning classifiers. Where possible, *gumpy* leverages existing and well established scientific and numerical libraries such as NumPy [18], SciPy [19] and scikit-learn [20] to compute the classification results or to perform signal analysis. For instance, *gumpy*'s SVM classifier utilizes scikit-learn. However, *gumpy* preconfigures its classifiers with default parameters that were found to be suitable in typical BCI applications. In addition, *gumpy* can perform a grid search to tune their settings. One of *gumpy*'s core design principles is to allow users to easily extend its functionality, thereby facilitating usability, customizability and collaborative development. The latter is further enabled using our public git repository at `https://github.com/gumpy-bci` through which we solicit the community to contribute feedback and code. In addition, the website `http://gumpy.org/` provides an API reference and usage examples in the form of Jupyter notebooks.

3.2. Gumpy's experimental paradigms

3.2.1. Classic motor imagery movements Gumpy provides a cue-based screening paradigm to record classic motor imagery; namely the imagination of the movement of left hand, right hand or both hands as shown in Figure 2. At predefined times, the screen displays a cue in the form of an arrow pointing either left, right or both ways. The participant has to perform a hand movement imagination accordingly.

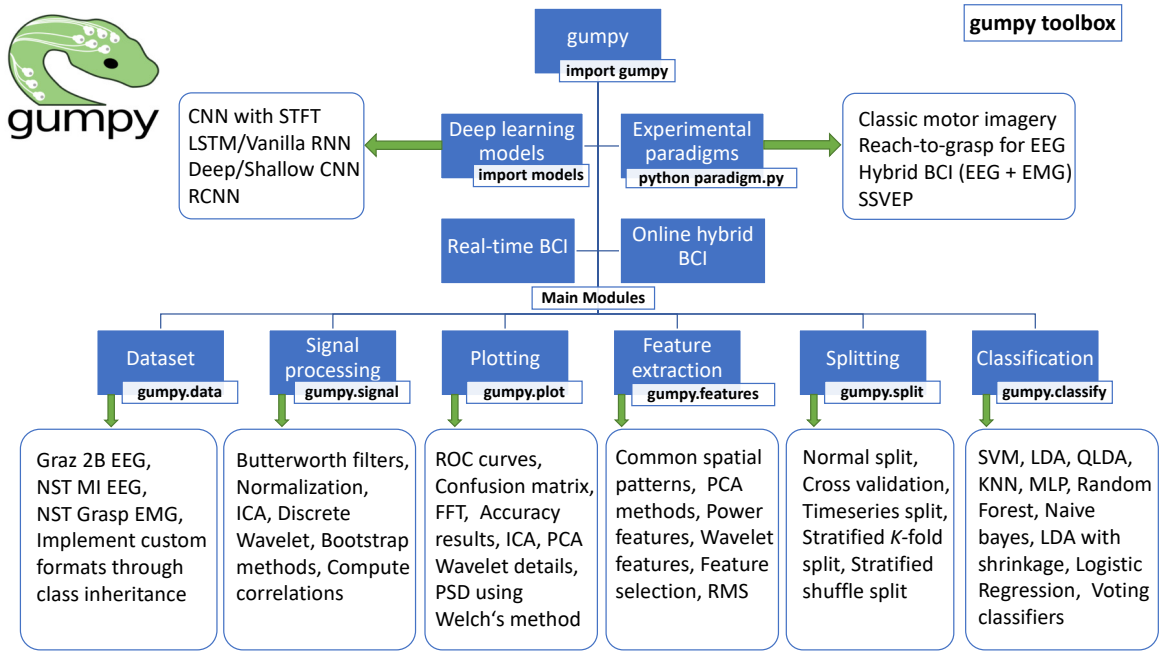

Figure 1: Overview of *gumpy* toolbox modules and functions.

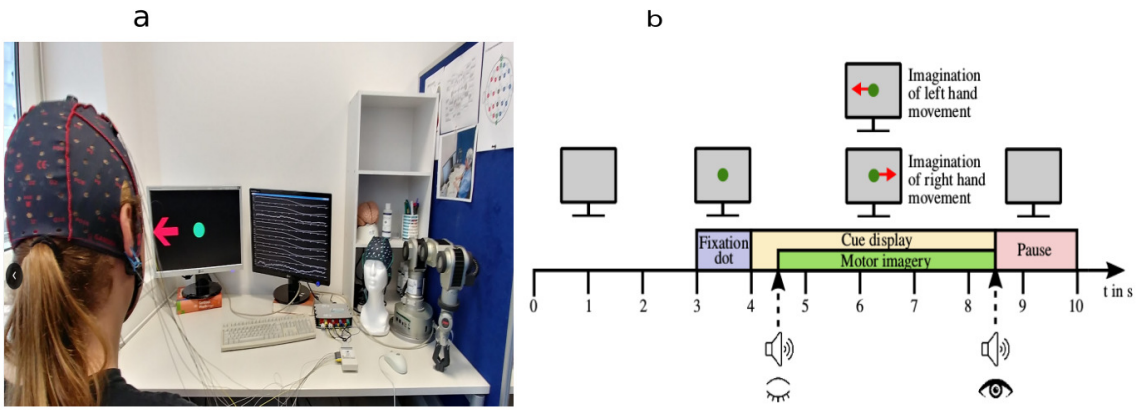

Figure 2: Illustration of recording paradigm for three motor imagery EEG data acquisition. (a) Photograph of a recording session. (b) Outline of the designed recording paradigm.

3.2.2. Reach-to-grasp motor imagery movements Gumpy incorporates a paradigm to record EEG reach-to-grasp movements imagination of six different objects placed on a shelf with fixed positions as shown in Figure 3. The subject is asked to imagine a reach movement by bringing the cursor (square) toward one of the six center-out target locations (up-left, up-right, center-left, center-right, down-left, down-right). Once the square hits the target, it turns red which triggers the participant to now imagine performing a grasping movement on that specific target.

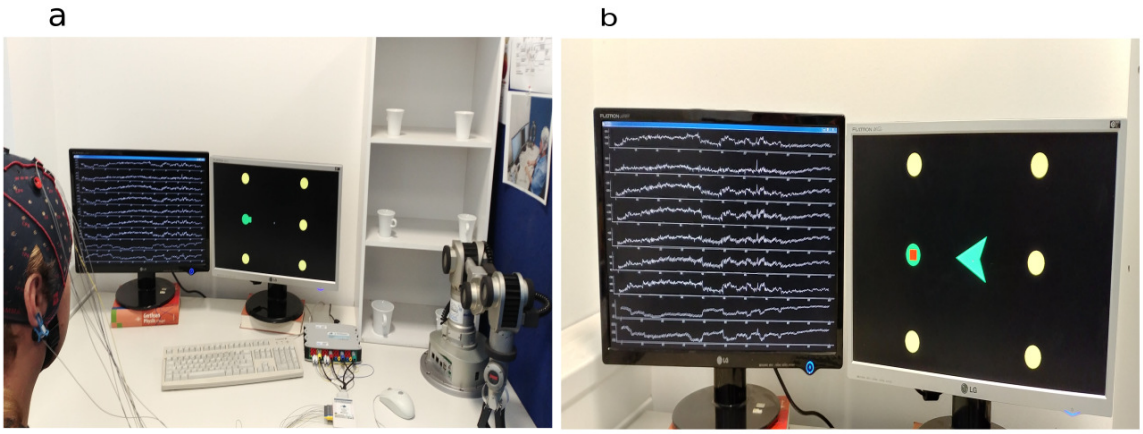

Figure 3: Illustration of recording paradigm for reach-to-grasp movements. (a) Display requests subject to imaginatively reach for the mid-left cup in the shelf. (b) Subject is requested to imagine grasping the center-right cup.

3.2.3. Grasp poses and related finger forces from surface EMG signals A special experimental paradigm was designed to record sEMG signals from the forearm during four different hand movements (2-digit grasp, 3-digit grasp, fist, hand open) as shown in Figure 4 with two possible force levels (high, low). Strain gauge sensors placed on the fingertips measured the applied grasping force [30] .

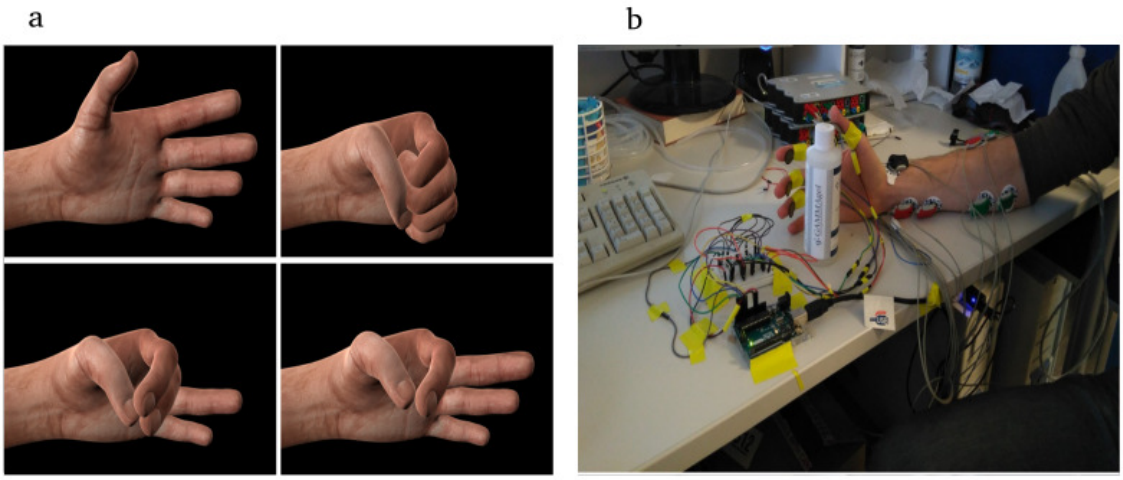

Figure 4: EMG recording paradigm. (a) Different hand gesture renderings prompting subjects. (b) Recording setup of EMG signals during grasp movements.

3.2.4. Gumpy-SSVEP paradigm The SSVEP paradigm consists of four flickering checkerboards blinking at different frequencies (13, 15, 17 and 19 Hz), as shown in Figure 5. The subject has to focus on one of the flickering checkerboards in order to evoke an SSVEP response. Simultaneously, EEG signals recording from O1, OZ and

O2 electrodes were performed. The paradigm was implemented using PyGame [31], a gaming-oriented Python library for graphical user interfaces. It requires a monitor supporting sufficiently high (or dynamic) refresh rates.

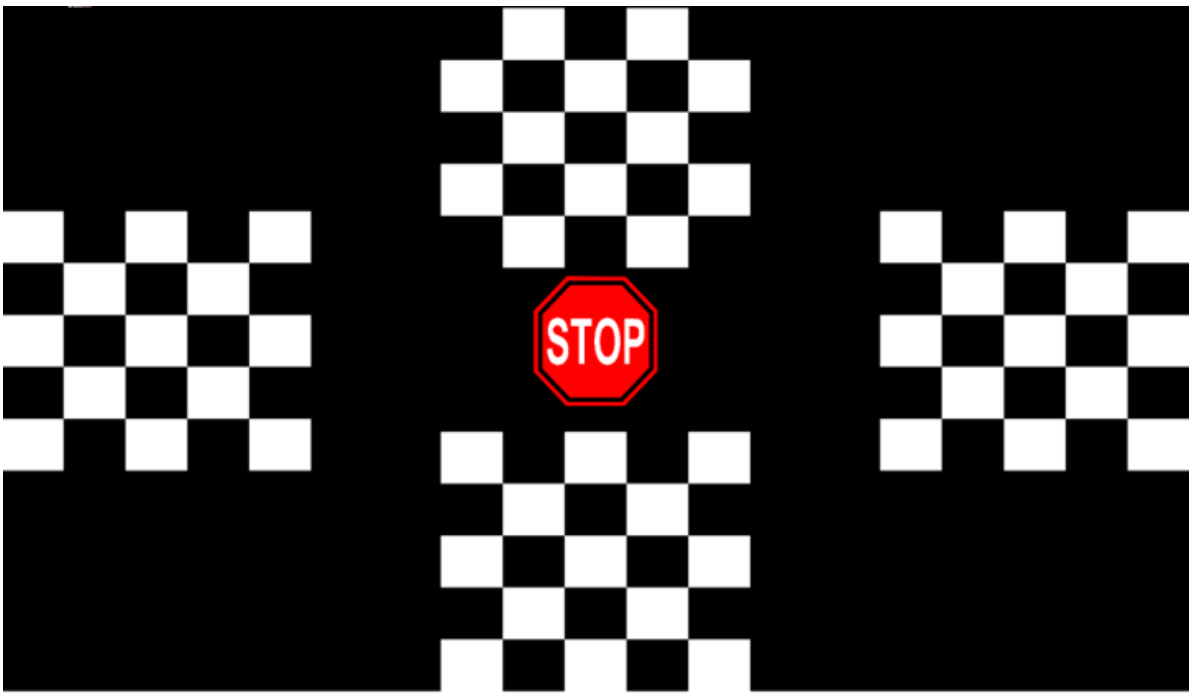

Figure 5: Illustration of the recording paradigm for SSVEP.

3.2.5. Gumpy's experimental paradigm for real-time hybrid BCI The hybrid BCI paradigm allows end users to perform online hybrid BCI experiments. For instance, this paradigm was used to perform a sequential hybrid BCI task, where the subject was asked to imagine left or right hand movement imagination and execute thereafter the same imagined movement. For that, a simultaneous recording of EEG and sEMG signals was performed using two synchronized g.USBamp devices. Signals were sampled at 512 Hz and the LSL was used for data acquisition in a master-slave communication fashion. It should be noted that the developed paradigm could be used to simultaneously collect data from other devices (e.g. Myo armband [32] and the g.USBamp) and could be easily modified to acquire other types of biosignals. A detailed documentation of the hybrid paradigm as well as the developed code are made publicly available within *gumpy* under `https://github.com/gumpy-bci`.

3.3. Gumpy's deep learning module

Despite the numerous successful applications of deep neural networks [10], the development of deep learning methods in the BCI field is still quite rare [12]. In this section, we describe *gumpy*'s deep learning module, which is based on Theano [33] and Keras [34], as well as different implemented and available network architectures.

3.3.1. Recurrent neural networks (RNN) Recurrent neural networks and particularly long short-term memory (LSTM) have been used successfully to model the temporal characteristics of diverse non-stationary and non-linear time-series signals. Likewise, such methods should be applicable to EEG data as well [35]. Gumpy makes LSTMs and other recurrent architectures like vanilla RNN and recurrent convolutional neural networks (RCNN) readily available and provides well-documented example code. The architecture of the LSTM algorithm distributed with the initial *gumpy* release is shown in Figure 6. It consists of one LSTM layer consisting of 128 cells and an input layer where E represents the electrode channels, T represents the number of samples in every channel and K the number of output classes.

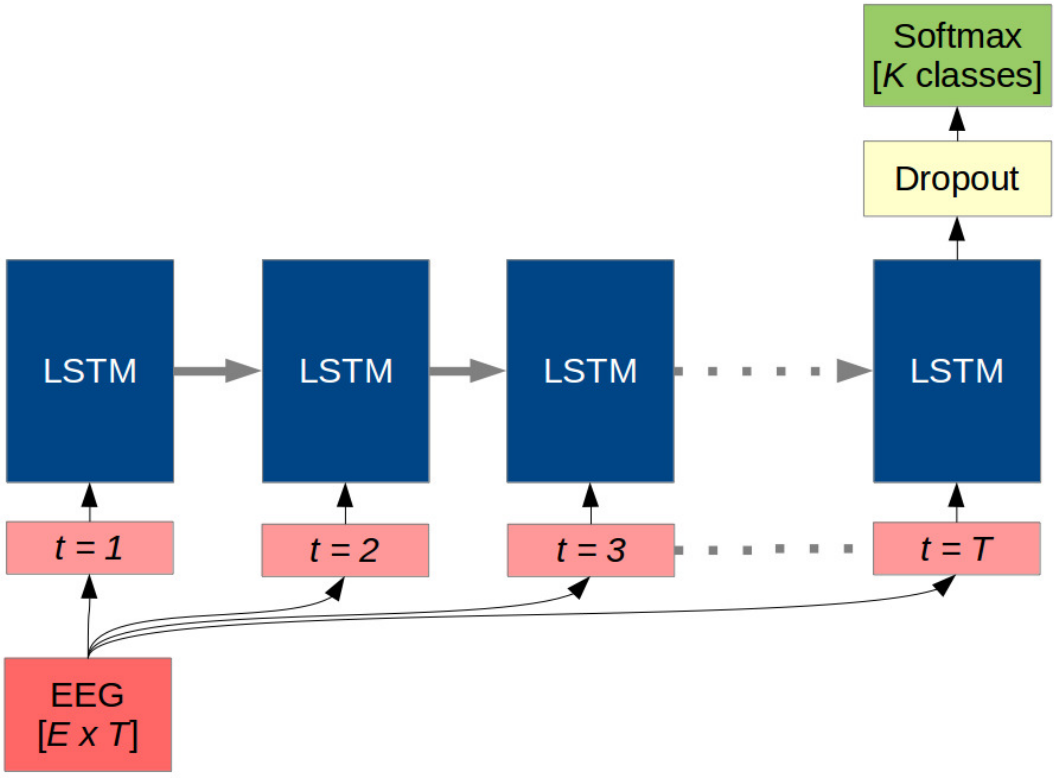

Figure 6: Implemented LSTM architecture.

3.3.2. Convolutional Neural Network (CNN) The proposed CNN model architecture is illustrated in Figure 7. The network architecture is inspired by CNNs used in the ImageNet competition, such as VGGNet [36] and AlexNet [37]. It uses stacked convolutional layers with decreasing size and increasing number of filter kernels in deeper layers. After each convolutional layer, batch normalization is applied to reduce covariate shift in intermediate representations and improve robustness. The actual spectrogram representation of the EEG signal is computed in the first layer and used as an input to

the model. A more detailed description of the CNN architecture and its implementation will be provided elsewhere [38].

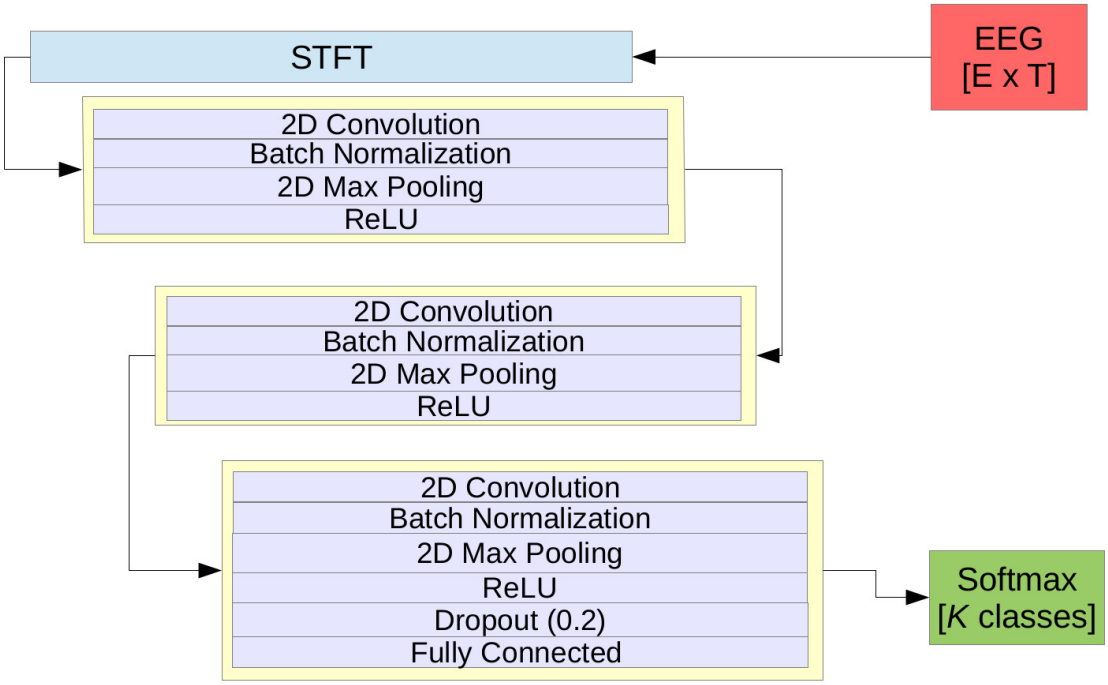

Figure 7: The proposed CNN architecture, where E is the number of electrodes, T is the number of timesteps and K is the number of classes.

4. Offline analysis case studies

In this section, we show how to use *gumpy* to perform offline analysis of EEG/EMG data. As a result, researchers can easily reproduce the obtained results using our provided Jupyter notebooks, our freely available EEG/EMG recorded data or the EEG dataset 2b from BCI competition IV [39] as well as *gumpy*'s available experimental paradigms.

4.1. Decoding of two motor imagery movements from Graz 2b EEG signals

We used *gumpy*'s signal and classification developed modules to process and classify an existing EEG dataset known as 2b EEG dataset from "BCI Competition IV" [40]. The source code (Jupyter notebooks) utilized in these offline examples are freely available under `http://gumpy.org/`.

4.1.1. Standard Machine learning techniques Three feature extraction methods, i.e. logarithmic band power (BP) [41], common spatial patterns (CSP) [42, 43] and discrete wavelet transform [44], have been investigated and tested. In general, CSP features maximize the pairwise compound variance between two or more classes in the least square sense, whereas wavelet features provide an effective time-frequency representation

of non-stationary signals [44]. We wish to emphasize that those feature extraction methods have been advocated by BCI researchers in [7]. After extracting discriminative features, `gumpy.features.sequential_feature_selector` was used to automatically select a subset of features in the feature space using the sequential feature selection algorithm (SFSF) [45]. The `gumpy.split` module provides several methods for splitting data. Herein, we used the hold-out strategy by splitting the dataset into 80% for training and 20% for test using the `gumpy.split` module. A 10-fold cross validation was performed on the training set to select the best features using six different classifiers from the `gumpy.classification` module. Afterwards, the new generated subsets based on the selected features were fed into each classifier and new predictions were made on the testing dataset. Furthermore, we wish to mention that the classification module incorporates a voting classifier, which employs an ensemble of classifiers to "vote" using their respective results. Finally, it should be noted that `gumpy.classification` can also perform a grid search to select the best hyper parameters for SVM and random forest classifiers for a given k-fold cross validation. Noticeably, BP slightly outperforms the other two feature extraction methods and provides overall better results across the different nine subjects. The obtained classification results with the BP feature extraction method with six different algorithms including the voting classifier are shown below in Figure 8. Overall, the obtained results from individual subjects show inter-

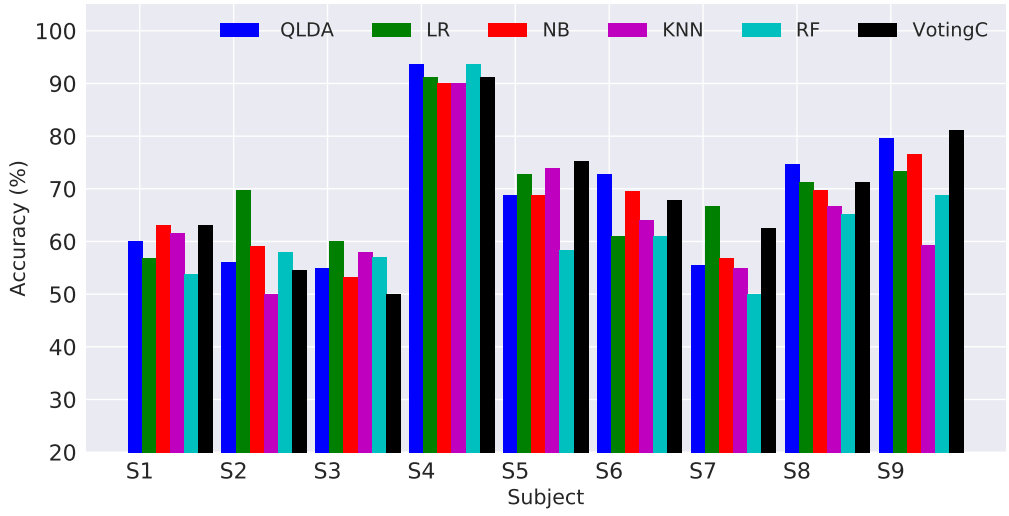

Figure 8: Accuracy results obtained for individual participants using the BP features and six different machine learning classifiers, namely quadratic LDA (QLDA), logistic regression (LR), naive bayes (NB), k-nearest neighbors (KNN), random forest (RF) and the voting classifier (VotingC).

and intra-subject variability. According to their performance, the nine participants could be classified into three categories: Bad participants are S1, S2, S3 and S7 with a classification accuracy between 60 to 70%, good participants are S5, S6, S8 and S9

with a classification accuracy between 70 to 82%, and an excellent participant S4 with an average classification accuracy of 93.75%. It is worth noting that a comparable performance was obtained with the CSP features. A Jupyter notebook showing how to use the three different feature extraction methods with the different available classifiers, is made publicly available under `https://github.com/gumpy-bci`.

4.1.2. Deep learning techniques Two deep neural network algorithms for motor imagery classification using the `gumpy.deep_learning` module were investigated and tested: convolutional and recurrent neural networks. Firstly, an LSTM model with only one hidden layer consisting of 128 LSTM memory cells was tested. To assess the model's capability of autonomously learning discriminative features, only raw EEG signals were fed into the algorithm. The large number of parameters of the LSTM model makes the model prone to overfitting. A dropout layer with a deactivation rate of 0.05 between the output and the LSTM layer partially mitigates this problem. Second, a CNN algorithm was implemented and tested. Recorded EEG signals were first cropped into short, overlapping time-windows. Thereafter, a fast Fourier transform (FFT) was performed on each crop, assuming stationarity in short time-frames. Spectrograms from three electrodes C3, C4 and Cz in the frequency band of 25-40 Hz were computed and used as inputs to our proposed CNN algorithm. Parameters were set to n = 1024 FFT samples and a time shift of s = 16 time steps between STFT windows. For each of the nine participants, a stratified 5-fold cross-validation was applied. Four folds were used for training and validation (90 % training, 10 % validation) and the last fold was used for testing. Finally, we point out that early stopping [46] was used to avoid overfitting. That means the model is trained until the minimum of the validation loss is found and then tested on the test data to measure its generalization capabilities. Interestingly, the obtained results with the CNN model outperformed the state of the art results on the same dataset, which were obtained with classic methods. However, LSTM results were similar to those obtained with traditional methods (e.g. quadratic LDA) as shown in Figure 9. An intuitive reason of that could be the limited amount of training data. As a result, reducing model complexity by decreasing the number of cell memories would be a promising solution to improve the developed algorithm. After validating the offline results, we wish to mention that the testing phase was done online and a successful real-time control of a robot arm was performed using the trained proposed CNN model as shown in the supplementary video in the supplementary materials section 7.

4.2. Decoding of natural grasps from surface EMG signals

Making a prosthetic hand grasp an object precisely and effortlessly is a crucial step in prostheses design [47]. Additionally, dexterous grasping of objects with different shapes and sizes seems to be a big challenge in today's prostheses. In this section, we demonstrate the usage of *gumpy* to classify four movements (Fist grasp, 2-digit grasp, 3-digit grasp, hand open) with two different force levels (low, high). Data used in this

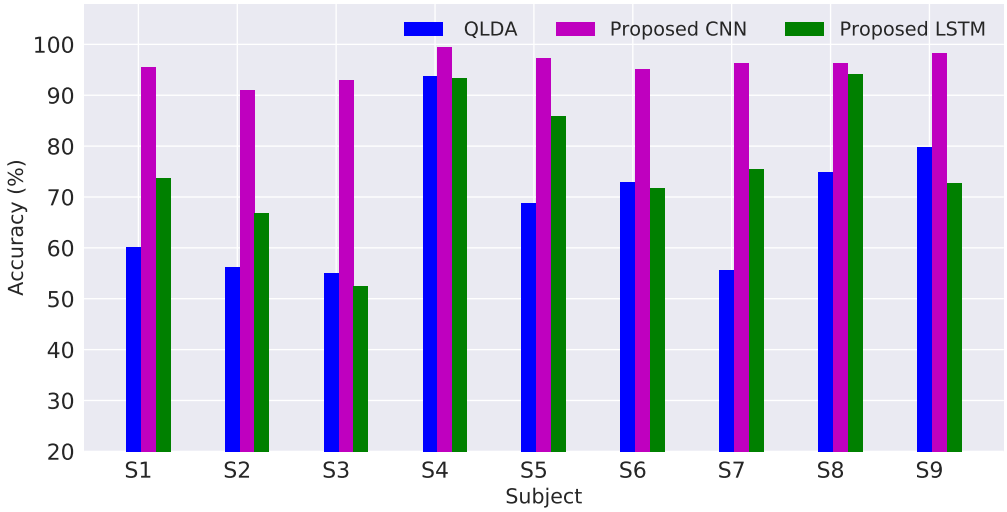

Figure 9: Accuracy results obtained for individual participants with QLDA, CNN and LSTM models.

example study were recorded at our lab and are made publicly available at *gumpy*'s website. Different steps for EMG processing using *gumpy* are described below.

Filtering EMG signals were band-pass filtered between 20 and 255 Hz and notch filtered at 50 Hz using the `gumpy.signal` module. Feature extraction and normalization: Filtered EMG signals were analyzed using 200 ms sliding time windows with 50 ms overlapping [48]. The length of the sliding window was chosen for the purpose of allowing real-time control. For each interval, the computed mean of the signal was subtracted and divided by the standard deviation. Besides, the resulting EMG signals were normalized by the maximum voluntary isometric contraction (MVIC) magnitude. Thereafter, the root mean square (RMS) was computed in each time window and fed into the classifier. We wish to stress that we used the same feature extraction method to classify each type of the associated force level (low, high).

Feature selection and Classification Herein, the SFFS algorithm was used to select a certain number of features in the k range (10, 25). Different classifiers were used to predict one of four possible hand poses and one of the two force levels. Offline results using SVM with 3-fold stratified cross validation are illustrated in Figure 10. Obtained prediction results during the real-time test, are presented in the next section. The validation accuracy for three different subjects were 82% ($\pm$4%) for posture classification and 96% ($\pm$3%) for force classification. It should be noted that those results were obtained after performing three-fold cross validation using *gumpy*'s validation module.

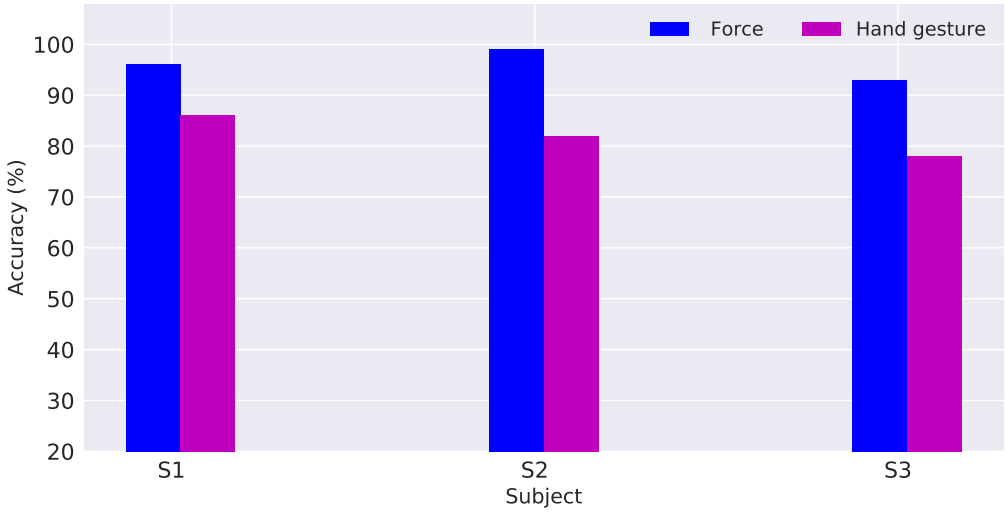

Figure 10: Obtained results for hand posture and force classification with 3-fold cross-validation.

5. Gumpy real-time applications

Aside of the offline capabilities, *gumpy* can be used in an online fashion to perform real-time experiments such as robot arm control using SSVEP, online EMG decoding and real-time control of robots using EEG signals. All the real-time *gumpy* case studies as well as the developed real-time experimental paradigms are made available under `https://github.com/gumpy-bci/gumpy-realtime`. Importantly, these case studies can be easily modified or extended by *gumpy* end-users to suit their specific applications.

5.1. Real-time Robot Arm Control using SSVEP-based BCI

In this section, we further test and validate *gumpy*'s real-time capabilities by online detection and classification of SSVEP signals for a robot arm control. SSVEP are brain events measured after a visual flickering stimulation of a frequency between 3.5 Hz and 75 Hz. They appear as a peak in the frequency spectrum of the EEG signals recorded over the primary visual cortex at the respective stimulus frequency [28]. The *gumpy* SSVEP paradigm described previously in section 3.2.4 was used for data recording. During the live experiment, the subject had to focus on one of the four displayed checkerboards flickering at different frequencies. Power spectral density (PSD) features from the electrodes O1, O2, and Oz over the occipital lobe were extracted, normalized and a principal component analysis (PCA) was performed to reduce the dimensionality. A random forest classifier was trained offline on recorded data collected from four different subjects (3 male, 1 female). A 5-fold stratified cross validation was performed to evaluate model performance and to tune hyper-parameters. Afterwards, the trained random

forest classifier was used in the testing phase to perform an online classification, where new predictions on the test data were performed. Thereafter, a command was sent to move a six degrees of freedom (6-DoF) robot arm in four different directions according to the position of the detected flickering object. A flowchart of this SSVEP project is shown in Figure 11. In addition, a supplementary video of this work, which shows a successful real-time robot arm control using SSVEP-based BCI is available in the supplementary materials section 7.

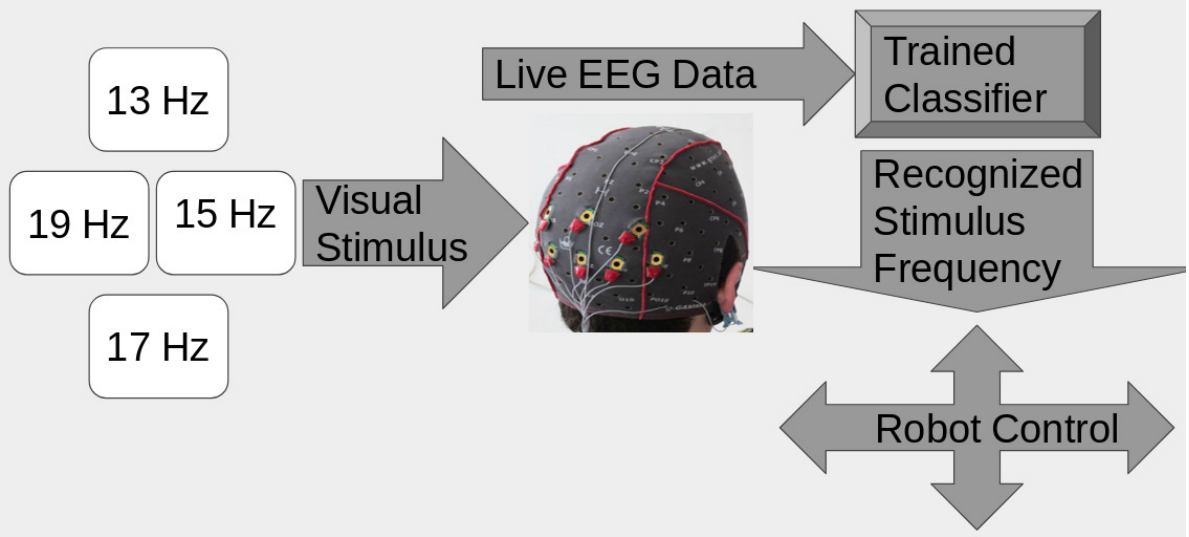

Figure 11: SSVEP project flowchart.

5.2. Real-time prosthetic hand using surface EMG signals

Herein, we describe the online decoding of three grasp poses, namely fist grasp, 2-digit grasp and 3-digit grasp. The offline analysis and processing described previously in section 4.2 were used. The developed algorithm was tested on two healthy subjects. 72 trials (24 for each posture) were first acquired to train the model. Thereafter, new 30 online trials (10 per posture) were used for online testing. The number of offline trials used for model training has been reduced in retrospective analysis to evaluate the effect of the training data size on the online classification accuracy as shown in Figure 12. It should be noted that a 3-fold cross validation was used to train the (offline) model in the first place. Figure 12 shows that with 72 offline trials, an accuracy of 82% and 92% was reached for S1 and S2, respectively. Overall, it is clear that the accuracy could be even further improved by increasing the number of training trials. However, by using 24 trials for each posture, a good compromise between duration of training time and accuracy of training was found. A supplementary video of this work, which shows a successful real-time prosthetic hand control using sEMG is available in the supplementary materials section 7.

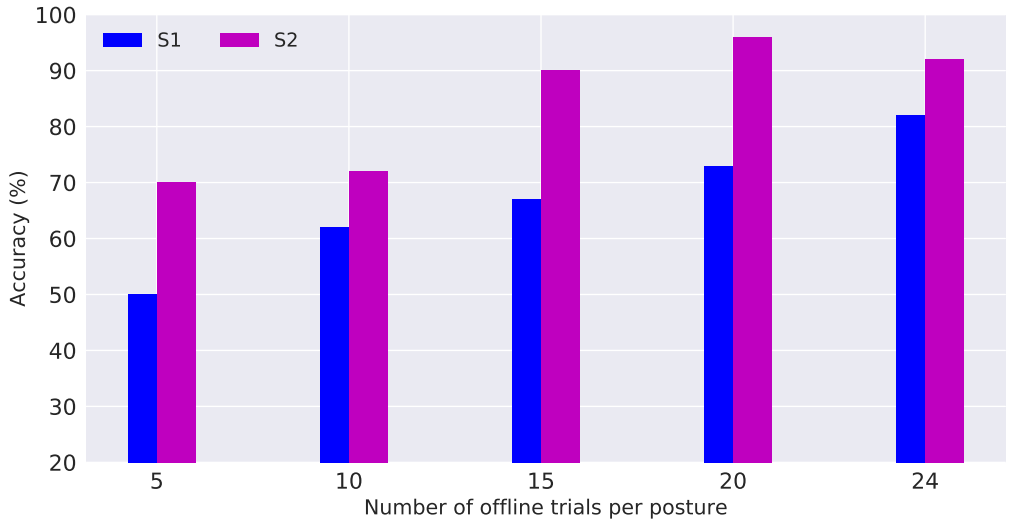

Figure 12: Online Accuracy of EMG classification without force.

5.3. Online hybrid BCI using EEG and surface EMG for reach-to-grasp movements

In this section, we present a case study for the hybrid BCI approach, where the decoding of motor imagery (MI) movements from EEG (Section 4.1) was combined with posture classification from sEMG (Section 5.2) in a sequential fashion. For that, 2-MI movements, namely left and right imagined hand movements and three classes of hand postures (fist, 2-finger pinch and 3-finger pinch) were decoded. Classic machine learning and deep learning approaches were combined to perform an online decoding. In this example study, the online mode was designed to perform reach-to-grasp movement decoding, where a KUKA robot arm [49] was controlled by MI signals (reach movement) whereas a prosthetic hand was controlled using sEMG signals (grasp movement) in a single online experiment. One benefit offered by combining EEG and EMG [50] is the low latency provided by EEG when decoding reach movements as well as the rich spectro-temporal information that can be decoded from sEMG when classifying complex grasp movements [50, 48]. In this example study, the LSL was used to synchronize different data streams (EEG, EMG) and the temporal procedure was arranged as a state machine. During the offline recording, the program alternates between two states, which execute the tasks related to EEG and EMG experiments. This means the participant performed the EEG experimental paradigm first. Thereafter, the EMG experimental paradigm was performed. This procedure was repeated for a defined number of offline trials, for instance 72 as was shown in the online EMG experiment in section 5.2. After completion of the offline experiments, the program enters a state, where the model of posture detection was trained based on the offline recorded EMG data whereas the MI pre-trained model was retrained based on the offline EEG data. It should be noted that the pre-trained model can be either a CNN or a standard machine learning classifier,

depending on the end-user's configuration. Afterwards, the program enters the online phase, which consists of three states (EEG, EMG and classification). These states are performed sequentially for a defined number of online trials. Likewise, the EEG state was first performed and was followed by the EMG state. As a result, data were classified and the robot arm as well as the prosthetic hand were controlled to perform a reach-to grasp movement as shown in Figure 13. It is worth noting that different experiments investigating the aforementioned hybrid approach are currently conducted and results will be reported in another scientific paper.

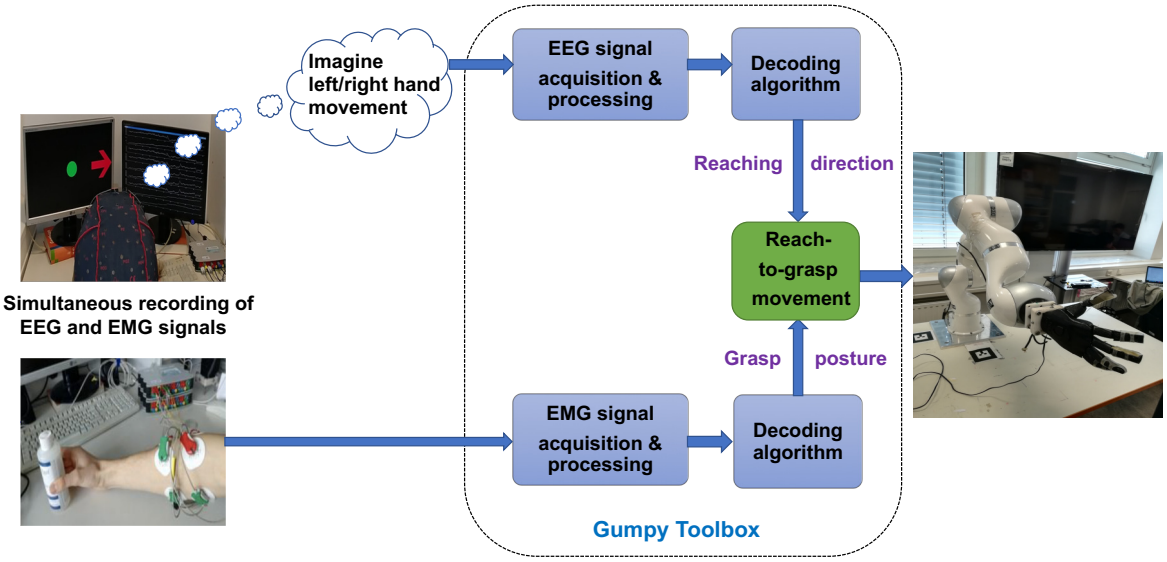

Figure 13: The proposed hybrid BCI experiment for reach-to-grasp movements decoding.

5.4. Live generation of spectrograms

In this section, we show how to use *gumpy* to generate and stream spectrograms, which could be used later on for different online applications. Generally, spectrograms are generated from data within a circular buffer that stores a predetermined number of samples up to the most recent one. The capacity of this buffer depends on the parameters used for the short-time Fourier transform (STFT), namely the window length and the overlap between consecutive windows. The window length is chosen as a compromise between frequency resolution in lower frequency bands and time resolution in higher ones. Prior to the STFT's application, the data are passed through the filter bank to ensure consistency in the signal range over all spectrograms. The training of a suitable network is realized with data augmentation methods, which mimic the live processing, so that the network is presented with similar data throughout training and real-time application. The live generation system has been tested for frame rates up to 128 Hz on

a PC with a 2.8 GHz quadcore CPU, showing a stable performance throughout. The source code of this live interface is available in the gumpy-realtime repository. Figure 14 and 15 summarize the whole process of live spectrograms generation. As shown in the video, a noticeable delay ($\sim$ 1.4 sec) was experienced when performing the real-time experiment. This delay can be attributed almost entirely to the CNN processing. Hence, modern hardware accelerators like NVIDIA TensorRT [51], Intel Movidius NCS [52] or even IBM TrueNorth [53] could reduce the latency drastically and provide much higher throughput for our developed deep learning models than the standard PC we have employed.

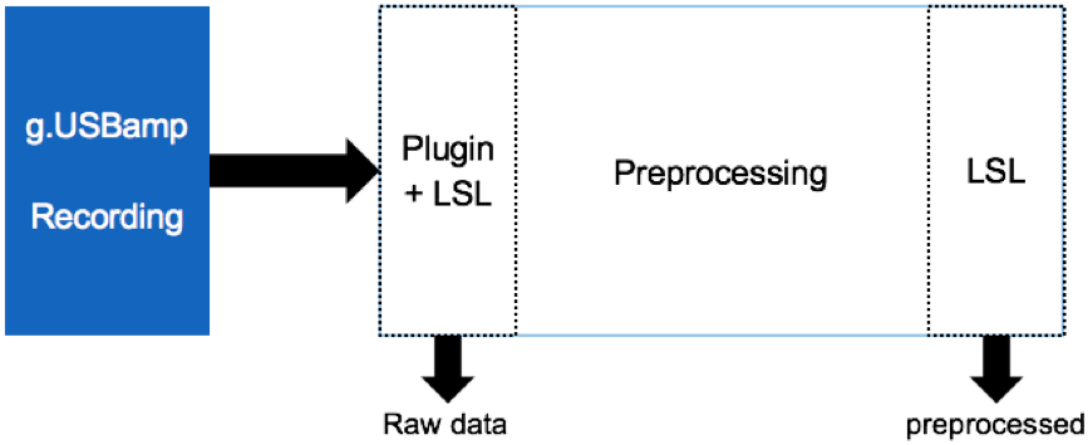

Figure 14: Data streaming via LSL.

6. Discussion and Conclusion

6.1. Gumpy toolbox advantages

In this paper, we unveiled *gumpy*, a free and open source Python toolbox for BCI applications. Gumpy includes a wide range of visualization, processing and decoding methods including feature selection algorithms, classic machine learning classifiers, voting classifiers and several deep learning architectures. Additionally, the toolbox is not only limited to EEG signals, but it can be used to interpret sEMG signals as well, hence spurring the usage of hybrid BCI concepts. Furthermore, *gumpy* provides a turnkey solution to perform online BCI experiments by providing several experimental paradigm examples including SSVEP, classic motor imagery movements, reach-to-grasp movements, EMG grasping tasks and online hybrid BCI experiments. In the previous sections, we demonstrated the usage of *gumpy* with two showcase examples for offline analysis using an existing EEG dataset and new EMG data recorded at our lab. Similarly, *gumpy*'s real-time capabilities were shown through the control of a robot arm using SSVEP-based BCI and the real-time control of a prosthetic hand using sEMG.

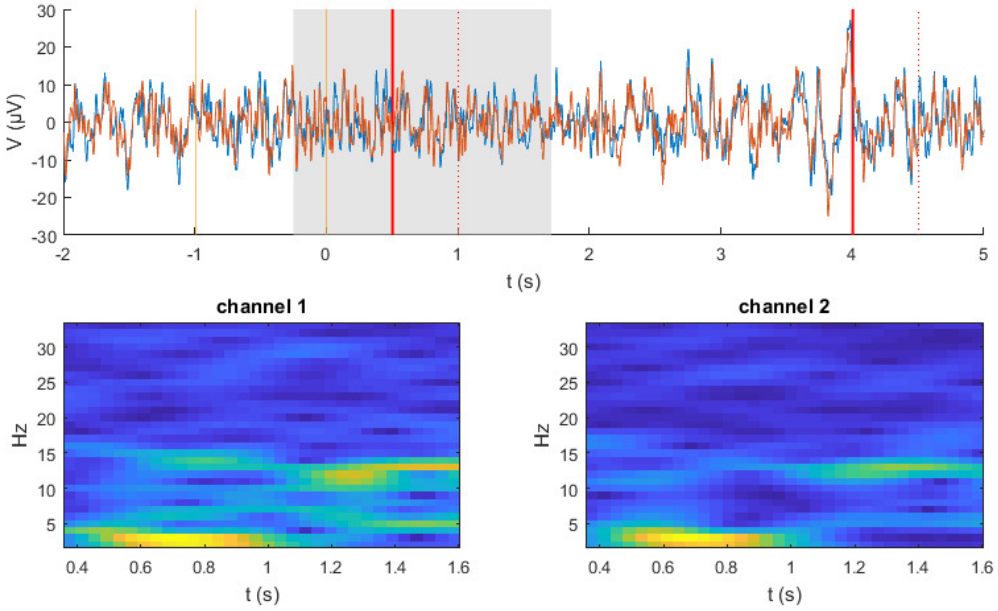

Figure 15: A frame of a live stream. Top: Filtered signal during a trial. Blue and red traces illustrate channel 1 and channel 2, respectively. Vertical lines indicate visual (orange, arrow at t = 0 s) and acoustic cues (red). Bottom: Generated spectrograms from data within the grey rectangle shown above.

More relevantly, *gumpy* includes different deep learning models such as CNN, RCNN and LSTM which were developed and tested in this paper to classify sensory motor rhythms from EEG signals. Interestingly, not only a reproducibility of previous results was achieved with gumpy but also some of the results (e.g. section 4.1) outperformed state-of-the art results on the same datasets. Thus, gumpy could foster the development of more robust EEG/EMG decoding algorithms and open new avenues in ongoing hybrid BCI research. Finally, it is important to highlight that different BCI research groups are now testing the toolbox and many students have already worked with it. Most of the students managed to master the use of the toolbox in less than a week.

6.2. Future development of gumpy toolbox

Despite the considerable number of functions, algorithms and experimental paradigms that *gumpy* provides, further processing methods are under development. Particularly, developing an experimental paradigm for error-related potential (ErrP) recording as well as providing a case study for ErrPs decoding would be of utmost importance for BCI researchers [54, 55]. Likewise, a P300-based BCI speller paradigm is still missing and should be added to *gumpy*'s experimental paradigms. Moreover, some of the widely-used techniques in BCI research, such as source localization [56] and connectivity analysis [57] should be integrated within the *gumpy* toolbox in future developments. Aside from that, it would be important to include channel selection techniques [58] as well as other classification methods to the toolbox, such as Riemannian geometry-based

classification [59] and restricted Boltzmann machines [60], which have been advocated by BCI researchers [7]. Moreover, in addition to the proposed sequential architecture in section 5.3, it would be important to test the simultaneous hybrid BCI, where EEG and EMG are fused to yield one control signal. This can be done by merging classification probabilities of EEG and EMG using Bayesian fusion techniques [5]. Furthermore, as *gumpy* was solely tested with EEG and EMG signals, performing more analyses with other human data, such as fMRI, ECoG or MEG could further validate the usefulness as well as the applicability of the toolbox, thereby spur the use of *gumpy* in other BCI applications. Last, we wish to highlight that some other example studies investigating the fusion of different multimodal signals [58] are now under development. Interesting works proposed before by Y. Li *et al.* about combining P300 and motor imagery [61] as well as combining SSVEP and P300 [62] present good sources of inspiration for developing and testing new multimodal BCI case studies. Along these lines, it would be undoubtedly important to investigate the combination of ErrP and EMG as has been recently proposed by J. DelPreto *et al.* in their novel work [63].

6.3. Conclusion

This paper presents and thoroughly describes *gumpy*, a novel toolbox suitable for hybrid brain computer interfaces. The overarching aim of *gumpy* is to provide a libre BCI software, which includes a wide range of functions for processing and decoding of EEG and EMG signals as well as classification methods with both traditional machine learning and deep learning techniques. The offline usage of gumpy is demonstrated with two different showcase examples. Firstly, gumpy is used to decode two motor imagery movements using a publicly available EEG 2b dataset from the BCI competition IV. Different feature extraction and classification methods have also been implemented and tested. Importantly, the obtained results using the *gumpy* CNN algorithm showed some improvement compared to obtained state-of-the art results on the same dataset. Furthermore, gumpy is also used to decode different grasp poses from our recorded *gumpy* signals. Additionally, we show gumpy's real-time capabilities within a successful robot arm control using SSVEP signals and a prosthetic hand control using sEMG. Last, we provide a case study where gumpy can be used to perform online hybrid BCI experiments. Overall, there are promising future trends for its use in various BCI applications. With gumpy, we envision to pave the way for a new phase of open source BCI research.

7. Supplementary Materials

- **Supplementary video 1** about real-time robot arm control using SSVEP-based BCI: `http://youtu.be/Dm-GGcImKjY`
- **Supplementary video 2** about EEG signals decoding using CNNs: `http://youtu.be/8hM7tOd7M7A`

- **Supplementary video 3** about prosthetic hand control using surface EMG signals: `http://youtu.be/igOEXpwfBZA`

8. Source code and documentation

The source code of gumpy toolbox is released under the MIT license and available with a detailed documentation at `http://gumpy.org`. In addition, we provide a tutorial-like overview of the toolbox using the python documentation generator Sphinx. With our provided Jupyter notebooks, we facilitate the usage of the toolbox and we give end-users insightful information how to adjust parameters in the toolbox.

9. Funding

This work was supported in part by Ph.D. grant of the German Academic Exchange Service (DAAD) and by the Helmholtz Association.

10. Acknowledgments

The authors would like to thank Othmane Necib, Rémi Laumont, Maxime Kirgo, Bernhard Specht, Daniel Stichling, Azade Farshad, and Sebastian Martinez, Constantin Uhde, Jannick Lippert and Chen zhong for technical assistance. We gratefully acknowledge the developers of Python, Theano, Keras, scikit-learn, NumPy, SciPy, LSL and other software packages that gumpy builds upon. Furthermore, we would like to thank Stefan Ehrlich for his helpful comments on the manuscript. Last, the authors would like to thank Prof. Dongheui Lee and Dr. Pietro Falco for fruitful discussions and for providing the prosthetic hand and the KUKA robot arm.

11. Conflicts of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships, which could be construed as a potential conflict of interest.

[1] R. G. S. Platts and M. H. Fraser. Assistive technology in the rehabilitation of patients with high spinal cord lesions. *Paraplegia*, 31:280–287, 1993.

[2] M. A. Lebedev and M. A. L. Nicolelis. Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation. *Physiol Rev*, 97(2):767–837, 2017.

[3] J. Meng, S. Zhang, A. Bekyo, J. Olsoe, B. Baxter, and B. He. Noninvasive electroencephalogram based control of a robotic arm for reach and grasp tasks. *Scientific Reports*, 6(2), 2016.

[4] Y. Li, J. Pan, F. Wang, and Z. Yu. A hybrid BCI system combining P300 and SSVEP and its application to wheelchair control. *IEEE Transactions on Biomedical Engineering*, 60(11):3156–3166, 2013.

[5] R. Leeb, H. Sagha, R. Chavarriaga, and J. d. R. Millán. A hybrid bci based on the fusion of EEG and EMG activities. *Journal of Neural Engineering*, 8:225–9, 2011.

[6] D. J. McFarland, W. A. Sarnacki, and J. R. Wolpaw. Electroencephalographic (EEG) control of three-dimensional movement. *Journal of Neural Engineering*, 7(3):036007, 2010.

[7] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger. A review of classification algorithms for EEG-based brain-computer interfaces: a 10 year update. *Journal of Neural Engineering*, 15(3):031005, 2018.

[8] Y. Rezaei Tabar and U. Halici. A novel deep learning approach for classification of EEG motor imagery signals. *Journal of Neural Engineering*, 14(1):016003, 2017.

[9] R. Ramadan and A. Vasilakos. Brain-computer interface: Control signals review. *Neurocomputing*, 223:26–44, 2017.

[10] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, 2017.

[13] C. Brunner, G. Andreoni, L. Bianchi, B. Blankertz, C. Breitwieser, S. Kanoh, C. A. Kothe, A. Lécuyer, S. Makeig, J. Mellinger, P. Perego, Y. Renard, G. Schalk, I. P. Susila, B. Venthur, G. R. Müller-putz, C. Brunner, C. Breitwieser, G. R. Müller-putz, C. Brunner, C. A. Kothe, S. Makeig, G. Andreoni, P. Perego, L. Bianchi, B. Blankertz, B. Venthur, S. Kanoh, J. Mellinger, Y. Renard, A. Lécuyer, G. Schalk, and I. P. Susila. BCI software platforms.

[14] C. A. Kothe and S. Makeig. BCILAB: a platform for brain-computer interface development. *Journal of Neural Engineering*, 10, 2013.

[15] A. Delorme and S. Makeig. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics. *Journal of Neuroscience Methods*, 134(1):9–21, 2004.

[16] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004.

[17] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. Hamalainen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7, 2013.

[18] S. V. Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13:20–23, 2011.

[19] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed: 2017-08-03].

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, M. Blondel O. Grisel, P. Prettenhofer, V. Dubourg R. Weiss, A. Passos J. Vanderplas, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] M. Grosse-Wentrup and M. Buss. Multiclass common spatial patterns and information theoretic feature extraction. *IEEE Transactions on Biomedical Engineering*, 8:1991–2000, 2008.

[22] V. Bastian, D. Sven, H. Johannes, H. Hendrik, and B. Benjamin. Wyrm: A brain-computer interface toolbox in python. *Neuroinformatics*, 13(4):471–486, 2015.

[23] B. Venthur and B. Blankertz. Mushu, a free- and open source BCI signal acquisition, written in Python. In *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, volume 2012, pages 1786–8, 08 2012.

[24] B. Venthur, S. Scholler, J. Williamson, S. Dahne, M. Treder, M. Kramarek, K. Múller, and B. Blankertz. Pyff—a pythonic framework for feedback applications and stimulus presentation in neuroscience. *Frontiers in Neuroscience*, 4:179, 2010.

[25] Y. Renard, F. Lotte, G. Gibert, M. Congedo, E. Maby, V. Delannoy, O. Bertrand, and A. Lécuyer. Openvibe: An open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments. *Presence*, 19(1):35–53, 2010.

[26] J. Sauvan, A. Lécuyer, F. Lotte, and G. Casiez. A performance model of selection techniques for P300-based brain-computer interfaces. In *CHI*, pages 2205–2208, 2009.

[27] Y. Li, J. Long, T. Yu, Z. Yu, C. Wang, H. Zhang, and C. Guan. An EEG-based BCI system for 2-d cursor control by combining Mu/Beta rhythm and P300 potential. *IEEE Transactions on Biomedical Engineering*, 57(10):2495–2505, 2010.

[28] Y. Wang, X. Gao, B. Hong, C. Jia, and S. Gao. Brain-computer interfaces based on visual evoked potentials. *IEEE Engineering in Medicine and Biology Magazine*, 27(5):64–71, 2008.

[29] Lab streaming layer. Available: `https://github.com/sccn/labstreaminglayer`.

[30] A. Gailey, P. Artemiadis, and M. Santello. Proof of concept of an online EMG-based decoding of hand postures and individual digit forces for prosthetic hand control. *Frontiers in Neurology*, 8:7, 2017.

[31] Pygame, [Online; accessed 2017-05-05]. Available: `https://www.pygame.org/`.

[32] M. Sathiyanarayanan and S. Rajan. MYO armband for physiotherapy healthcare: A case study using gesture recognition application. In *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–6, 2016.

[33] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[34] François Chollet et al. Keras. `https://keras.io`, 2015.

[35] M. Li, M. Zhang, X. Luo, and J. Yang. Combined long short-term memory based network employing wavelet coefficients for MI-EEG recognition. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 1971–1976, 2016.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105. Curran Associates Inc., 2012.

[38] Z. Tayeb, J. Fedjaev, N. Ghaboosi, C. Richter, L. Everding, X. Qu, Y. Wu, G. Cheng, and J. Conradt. Validating deep neural networks for online decoding of motor imagery movements from EEG signals. *IEEE access*, 2018. (Submitted).

[39] R. Leeb, C. Brunner, G. Mueller-Put, A. Schloegl, and G. Pfurtscheller. BCI competition 2008-Graz data set b. *Graz University of Technology, Austria*, 2008.

[40] R. Leeb, F. Lee, C. Keinrath, R. Scherer, H. Bischof, and G. Pfurtscheller. Brain-computer communication: Motivation, aim, and impact of exploring a virtual apartment. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(4):473–482, 2007.

[41] N. Brodu, F. Lotte, and A. Lécuyer. Comparative study of band-power extraction techniques for motor imagery classification. In *2011 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB)*, pages 1–6, 2011.

[42] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. Designing optimal spatial filters for single-trial EEG classification in a movement task. *Clinical Neurophysiology*, 110(5):787 – 798, 1999.

[43] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan. Filter bank common spatial pattern (FBCSP) in brain-computer interface. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2390–2397. IEEE, 2008.

[44] F. Sherwani, S. Shanta, B. S. K. K. Ibrahim, and M. S. Huq. Wavelet based feature extraction for classification of motor imagery signals. In *2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pages 360–364, 2016.

[45] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Lett*, 15(11):1119–1125, 1994.

[46] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning*, 11:625–660, 2010.

[47] E. Biddiss and T. Chau. The roles of predisposing characteristics, established need and enabling resources on upper extremity prosthesis use and abandonment. *Disability and Rehabilitation: Assistive Technology*, pages 71–84, 2009.

[48] I. Batzianoulis, S. El-Khoury, E. Pirondini, M. Coscia, S. Micera, and A. Billard. EMG-based decoding of grasp gestures in reaching-to-grasping motions. *Robot. Auton. Syst.*, 91(C):59–70, May 2017.

[49] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR lightweight robot arm -a new reference platform for robotics research and manufacturing. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, 2010.

[50] F. Artoni, A. Barsotti, E. Guanziroli, S. Micera, A. Landi, and F. Molteni. Effective synchronization of EEG and EMG for mobile Brain/Body imaging in clinical settings. *Frontiers in Human Neuroscience*, 11:652, 2018.

[51] NVIDIA Corporation. NVIDIA TensorRT. Available: `https://developer.nvidia.com/tensorrt`.

[52] Intel Corporation. Intel Movidius Neural Compute Stick. Available: `https://developer.movidius.com/`.

[53] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Cassidy andrew S, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[54] N. M. Schmidt, B. Blankertz, and M. S. Treder. Online detection of error-related potentials boosts the performance of mental typewriters. *BMC Neuroscience*, 13(1):19, 2012.

[55] R. Chavarriaga, A. Sobolewski, and J. del. R. Millán. Errare machinale est: the use of error-related potentials in brain-machine interfaces. *Frontiers in Neuroscience*, 8:208, 2014.

[56] M. G. Wentrup, K. Gramann, E. Wascher, and M. Buss. EEG source localization for brain-computer interfaces. In *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.*, pages 128–131, 2005.

[57] M. Hamedi, S. Salleh, and A. M. Noor. Electroencephalographic motor imagery brain connectivity analysis for BCI: A review. *Neural Computation*, 28(6):999–1041, 2016.

[58] T. Yu, Z. Yu, Z. Gu, and Y. Li. Grouped automatic relevance determination and its application in channel selection for P300 BCIs. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(6):1068–1077, 2015.

[59] M. Congedo, A. Barachant, and R. Bhatia. Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review. *Brain-Computer Interfaces*, 4(3):155–174, 2017.

[60] A. Fischer and C. Igel. An introduction to restricted boltzmann machines. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 14–36, 2012.

[61] Y. Li, J. Long, T. Yu, Z. Yu, C. Wang, H. Zhang, and C. Guan. An EEG-based BCI system for 2-d cursor control by combining Mu/Beta rhythm and P300 potential. *IEEE Transactions on*

Biomedical Engineering, 57(10):2495–2505, 2010.

[62] Y. Li, J. Pan, F. Wang, and Z. Yu. A hybrid BCI system combining P300 and SSVEP and its application to wheelchair control. *IEEE Transactions on Biomedical Engineering*, 60(11):3156–3166, 2013.

[63] J. DelPreto, A. F. Salazar-Gomez, S. Gil, R. M. Hasani, F. H. Guenther, and D. Rus. Plug-and-play supervisory control using muscle and brain signals for real-time gesture and error detection. In *Proceedings of Robotics: Science and Systems*, 2018.